CS 342 Design Document
Project 1 – Alarm Clock Implementation

Shrinivas Acharya
10010164

I went through the following link other than the pintos manual and course notes:
http://www.stanford.edu/class/cs140/projects/pintos/pintos.html

---- DATA STRUCTURES ----

Added to struct thread:

/* Member to store the ticks to sleep for each individual thread. */
int64_t ticks_to_sleep;

Declared the following list in thread.h:

/* list to store the sleeping threads */
struct list sleep_list;

The list was then initialised in thread.c in the function thread_init(), along with other lists.

---- ALGORITHMS ----

void timer_sleep(int64_t ticks)

In this function, I set the ticks for which the current thread is to sleep (thread_current() -> ticks_to_sleep) to be the argument ticks. Then I iterated the sleep_list (as directed in list.c) to insert the thread in the list in non-decreasing order of the thread's ticks_to_sleep. This allows me to keep the thread which is to be awakened earliest at the top of the list. Then the current thread is blocked.

static void timer_interrupt (struct intr_frame *args UNUSED)

This function is called after each timer interrupt. So in this function, I first iterated the sleep_list and decremented the ticks_to_sleep of each thread by one (the ticks are decremented only of the threads whose ticks_to_sleep are positive to start with). Then I started checking the list from top, and each of the threads whose ticks_to_sleep are less than or equal to zero removed from the list and then unblocked. This process is stopped as soon as any htread with a positive ticks_to_sleep is encountered, because the list is sorted and hence all further threads would also have positive ticks_to_sleep.

---- SYNCHRONIZATION ----
For synchronisation, as the codes here are of very small length, I have used interrupt disabling.

In the function timer_sleep, I disable the interrupts while inserting the thread in the list and blocking the thread.

The entire funciton timer_interrupt is run in interrupt disabled mode as directed in the link mentioned at the begning of the document. The reasoning behind this is that this is an external interrupt handler, and while such kind of interrupt service routine is being run, no other interrupts must be entertained.

---- RATIONALE ----
The algorithms implemented are very simplistic. The sleep list is firstly populated according to the ticks_to_sleep, so that while popping the threads, the list can be checked from the top, and only as long as the threads ticks are less than or equal to zero.

The improvement in this implementation as compared to the original implementation is that there is no more “busy waiting” in the threads which are sleeping.